MULTIMEDIA UNIVERSITY

TABLE NO: | | |

STUDENT ID NO: | | | | | | | | | | |

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2017/2018

TCP1101 – PROGRAMMING FUNDAMENTALS

(All sections / Groups)

9th Mar 2018
3.00 pm – 5.00 pm
(2 Hours)

Question	Mark
1	
2	
3	
4	

Total:

Examiner 1 Signature: _____

Examiner 2 Signature: _____

INSTRUCTIONS TO STUDENT

1. This Question paper consists of 12 printed pages (including cover page) with 4 Questions only.

2. Student must attempt **Question 1** and choose **THREE** out of the remaining four questions. All questions carry equal marks (10 marks). The distribution of the marks for each question is given.

3. Please print all your answers **CLEARLY** in the specific answer box provided for each question. Submit this question paper at the end of the examination.

Unless otherwise stated, every code segment provided in every question is contained within a main function, complete with the directive `#include <iostream>` and the statements `using namespace std;` and `return 0;`

QUESTION 1

(a) Write the output as the result of executing the following program. [4]

```
#include <iostream>
using namespace std;

int main(){
   int x = 6, y = 0, z = 6;

   {
        int x = 5;
        cout << x << endl;
   }
   cout << x++ << endl;

   if (x > z)
        cout << x << endl;
   else
        cout << y << endl;
   cout << z << endl;

   return 0;
}
```

Answer:

(b) Rewrite the following *for-loop* by using a *while-loop*. [4]

```
int num;
for (cin >> num; num < 0; cin >> num)
    cout << "Negative" << endl;
```

Answer:

(c) Write a C++ code segment by using the loop structure to produce the output pattern on the right. The user should be able to set the number of lines. In this example, the user sets the number of lines to four.
Assume that the value entered for N is always >= 3. [6]

Example 1:	**Example 1:**

When the user enters 6, the output pattern is:
`*`
`***`
`*****`
`*******`
`*********`
`***********`
	When the user enters 4, the output pattern is:
	`*`
	`***`
	`*****`
	`*******`

```
int N;
cin >> N;
// Write your code here:
```

(d) Draw a flowchart to accept a series of positive integers from the user, and then output the **sum of all odd numbers** entered by the user. User input terminates when any negative number is entered. [6]

Example 1:	**Example 2:**
`Enter integers: 20 3 18 1 -5` `Sum : 4`	`Enter integers: 15 30 4 7 3 -24` `Sum : 25`

Answer for 1 (d):

QUESTION 2

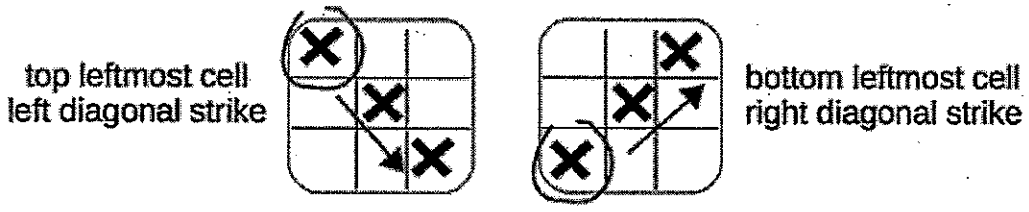
(a) For each question (a1) to (a6), in the **Output** column, write the output as a result of executing the code snippet in **Code** column. [6]

	Code	Output
a1	`char x[] = {'a', 'b', 'c'};` `cout << x[1];`	
a2	`char x[] = "ow";` `cout << x[1] << x[0] << x[1];`	
a3	`char x[] = "dec";` `cout << char(x[2] - 2);`	
a4	`string x = "abc";` `cout << x + "xyz";`	

a5	`int x[] = {3, 2, 1};`
	`for (int i = 1; i < 3; ++i)`
	`  x[0] += x[i];`
	`cout << x[0];`	

a6	`int x[] = {2, 1, 0};`
	`string y = "bad";`
	`for (int i = 0; i < 3; ++i)`
	`  cout << y[ x[i] ];`	

(b) Tic-Tac-Toe game is a two-player game played on a 3 by 3 board in which players compete to be the first to make a strike. A strike is a sequence of similar player symbols aligned horizontally, vertically, or diagonally. Examples of diagonal strike are as follows.



Write a program that accepts 9 inputs for the board and reports whether or not there is a diagonal strike. Each input is either a dash symbol ('-'), a capital letter 'X', or a digit zero ('0'). The dash symbol represents an empty slot, whereas capital letter 'X' and digit zero represents the symbol for player one and two respectively. The inputs are separated by either a whitespace or newline character. The program output should be similar to the examples given below. (NOTE: User inputs are in ***bold*** and ***italic.***) [7]

Example 1:	**Example 2:**	**Example 3:**
Enter a board: *X - O* *O X -* *- O X* There is a X diagonal strike.	Enter a board: *X - O* *- O X* *O O X* There is a O diagonal strike.	Enter a board: *- X O* *- - X* *X O -* No diagonal strike.

```
#include <iostream>
using namespace std;
int main()
{
      char board[3][3];
      cout << "Enter a board:\n";
      // Write your code here









      return 0;
}
```

(c) Complete the program below by defining the function **countOddAndEven()** that computes the number of odd and even digits in a given positive integer. The program output should be similar to the examples given below.
(NOTE: User input is made bold and underlined) [7]

Example 1:	**Example 2:**
Enter an integer: **0**	Enter an integer: **122423141**
Number of odd digits : 0	Number of odd digits : 4
Number of even digits: 1	Number of even digits: 5

```
#include <iostream>
using namespace std;

// Define the countOddAndEven() function here:









int main()
{
      unsigned int integer;
      cout << "Enter an integer: ";
      cin >> integer;

      int oddCount = 0, evenCount = 0;

      countOddAndEven(integer, oddCount, evenCount);

      cout << "Number of odd digits : " << oddCount << endl;
      cout << "Number of even digits: " << evenCount << endl;

     return 0;
}
```

QUESTION 3

You are required to create a **`Loan`** class.

This class has these private members:
ID : an integer representing a unique number assigned to a client
amount : a floating point number representing the loan amount
rate : a floating point number representing the interest rate of the loan
term : an integer representing the period of the loan in month

The public members are:
set : a void function that has an empty parameter list
payment : a function that has an empty parameter list, returns a floating point value
display : a void function that has an empty parameter list
Loan : a constructor with four parameters corresponding to the four private data members.

a) Write the definition of the **`Loan`** class. [4]

b) Implement the **`set()`** member function that will initialize the private data members of a **`Loan`** object by prompting the user for input. [4]

c) Implement the **`payment()`** member function that will compute the monthly payment for a **`Loan`** object, by using the following formula: [4]

$$monthly_payment = \frac{amount + (amount * (rate/100.0) * (term/12.0))}{term}$$

d) Implement the **`display()`** member function that will display all the member data of a **`Loan`** object. [4]

e) Implement a constructor with parameters that will initialize all the data members of a **`Loan`** object according to the values of the parameters. [4]

Sample code sections (to call the constructors and the member functions) and the relevant outputs are shown in the next page.

```
//Sample code_1
double monthly_payment;
Loan l(100123, 6590, 5.2, 24); //Calling the constructor
monthly_payment = l.payment();
cout<<"Loan Details: "<< endl;
l.display();
cout<<"The Monthly payment is: $"<< monthly_payment<<endl;
```

```
//Sample output_1
Loan Details:
Client ID        : 100123
Loan Amount      : $6590
Interest rate(%): 5.2
Period(months)   : 24

The Monthly payment is: $303.14
```

```
//Sample code_2
double monthly_payment;
Loan l;  //Calling the default constructor
l.set(); //Getting user input
monthly_payment = l.payment();
cout<<"Loan Details:"<<endl;
l.display();
cout<<"The Monthly payment is: $"<<monthly_payment<<endl;
```

```
//Sample output_2
Enter the Client ID: 201201
Enter the loan amount: 6000
Enter the annual interest rate(%): 8.5
Enter the term(months): 36

Loan Details:
Client ID       : 201201
Loan Amount     : $6000
Interest rate(%): 8.5
Period(months)  : 36

The Monthly payment is: $209.167
```

Answer:

.....continued for answer from previous page

QUESTION 4

(a) Write a function with the name **`getInt()`** to accept a user's input of an integer number. The function will accept no parameter but will return an integer value. The function will check the user's input, if it is not an integer it will clear the input buffer, display a warning message, and prompt the user for input again. If the input is an integer, the function will return the value. (Hint: the **`cin.clear()`** and **`cin.ignore()`** functions might be helpful in this.) [5.5]

(user's input are in *italic* and ***bold***)

Sample Run 1:

80

Sample Run 2:

```
apollo9
Wrong input. Please enter again: ice
Wrong input. Please enter again: w3
Wrong input. Please enter again: -4
```

Note:

Input is correct.
Value will be returned.

When Input is incorrect, displays error message, prompts for input again, until input is correct, and value will be returned.

Answer:

(b) Create a pointer version of the strlen function, call it **`myStrlen()`**. The function will receive a C string (a null-terminated character array) as a parameter, and will return the string length of it. You should use a pointer in the function to traverse the C string. You are not allowed to use any other predefined library functions here. [6.5]

Answer:

(c) Write the output generated by the following code: [4]

```
double x = 2.0;
double *ptr;

ptr = &x;

cout << x++ <<" "<< (*ptr)++ <<" ";
cout << x << " " << *ptr;
```

Answer:

(d) Within a C++ program, create a pointer to an integer, use that pointer to create a dynamic variable of the type int. Assign the value 88 to this dynamic variable. Lastly, destroy that dynamic variable and free up the memory occupied by it. [4]

Answer: